Trust Models in ICE-TEL

Andrew Young, IT Institute, University of Salford, UK
Nada Kapidzic Cicovic, COST Computer Security Technologies, Sweden
David Chadwick, IT Institute, University of Salford, UK

Abstract

Public key certification provides mechanisms that can be used to build truly scaleable security services, such as allowing people who have never met to have assurance of each other's identity. Authentication involves syntactic verification of a certificate chain followed by a semantic look at the policies under which the certificates were issued. This results in a level of assurance that the identity of the person to be authenticated is an accurate description of the person involved, and requires verifiers to specify who they trust and what they trust them to do. Two widely discussed mechanisms for specifying this trust, the PEM and PGP trust models, approach the problem from fundamentally different directions. The EC funded ICE-TEL project, which is deploying a security infrastructure and application set for the European research community, has described a new trust model that attempts to be equally applicable to organisation-centric PEM users and user-centric PGP users.

1 Introduction

The use of public key cryptography has made possible many developments in security technology. If each user can keep their private key completely private, and make their public key completely public then confidentiality, integrity, authentication and non-repudiation are possible in many different applications, removing many of the threats posed by the use of open networks. Making a public key public sounds like a relatively simple matter. However, in practice it introduces many new problems of its own. The problem is that all of the mechanisms commonly used for advertising public keys are insecure themselves. White pages directories, finger files, WWW pages and DNS are mechanisms that are commonly used or considered. However, it is not possible merely to store a public key in these mechanisms as the key can be modified in several ways that would breach security.

The solution to this problem is certification, where rather than distributing a public key, a public certificate is advertised. This is a package combining a user's identity, the user's public key, validity information and a digital signature appended by a third party (a **certification authority** (CA)) confirming their belief in the binding between the identity and the public key for the interval of the validity. If a user requiring someone else's public key knows something about the CA that signed the certificate, then they can determine if the certificate has been maliciously or accidentally modified.

This certification mechanism introduces the extra problem of **Trust Models**: what individual users are expected to know about these certificate authorities; who (if anyone) individual users have to trust in order to obtain useful results; and what mechanisms are needed to allow two users who have never previously met and have not prearranged the conversation to establish a useful level of assurance about the identity of each other. There are at least three well known models that give solution for this problem: ITU-T X.509 standard [1], RFC 1422 (commonly known as the PEM model) [2] and the PGP Web of Trust [3]. None of the models has met the needs of the general user community. X.509 defines completely general and theoretical model that allows arbitrary certification chains. RFC 1422 constrains this to make it tractable but requires considerable infrastructure to be established and functional before users can start to make meaningful use of the service. PGP trust model is based almost entirely on users mutual exchange of keys prior to the initiation of the first communication, and therefore is not applicable for complex organisations.

Within the ICE-TEL project [4], we have tried to create a trust model that is going to be acceptable both for individual users and arbitrarily complex organisations. This paper presents our results.

In section 2 we list our requirements for a useful and generally applicable trust model, and in section 3 we provide definitions for all of the terms we use. In the next three sections, we describe our trust model: first in general terms, then with regard to implementations, and finally some

pragmatic restrictions to ensure that implementations can be efficient. In section 7 we present some worked examples, and in section 8 we describe how the model can handle organic growth when requirements change with time.

2 Requirements

Based on experience gained previously with other trust models, the following requirements can be listed for the ICE-TEL trust model:

1. The trust model shall be capable of operating without the use of certificates or CRLs, through trusted exchange of public keys.
2. Where certificates are used, the trust model shall require the use of the X.509 standard v3 certificates and v2 CRLs expected to be ratified in 1997 [5]. Earlier versions of the certificate and CRL may be supported by implementations while they remain in widespread use, though these may not provide access to all facilities of the trust model. Proprietary certificate and CRL formats shall not be supported.
3. The trust model shall allow for the creation of security domains encompassing
   - single users
   - multiple users (small/simple organisations); and
   - arbitrarily complex organisations.
4. The trust model shall allow organic growth among the users and organisations; and shall allow security domains to grow, shrink and reorganise at any time with minimum inconvenience to the users within the domain and to people communicating with those users.
5. The trust model shall allow the administrator of a security domain to choose which other domains are to be trusted. There shall be no requirement that inter-domain trust is mutual and there shall be no points that any domain is required to trust. Inter-domain trust shall not be transitive.
6. The operation of the trust model as a whole shall not depend on the existence and operation of any single part of the deployed infrastructure. In particular, there need not be a central top level registration authority like the PEM IPRA [2].

3 Definitions

In this section, we define the key terms used in this paper, to ensure that there is no confusion with similar terms used elsewhere.

a) **key pair**
A key pair comprises the two complementary keys for use with an asymmetric encryption algorithm. One key, termed the public key, is expected to be widely distributed; and the other, termed the private key, is expected to be restricted so that it is only known to appropriate entities.

b) **user**
A user is a person or process that requires to make use of the trust model to obtain the public key of another person or process. All users are expected to have their own key pair, though this is not necessary for all applications. A user may have more than one key pair if required (for example to be used by the same person in multiple organisational roles).

c) **certification authority (CA)**
A CA is an entity who issues and revokes certificates according to an advertised certification policy (a CA may have multiple policies, for example a university may have different policies for staff and students). All CAs will have their own key pair.

d) **entity**
A CA can issue a certificate for a user or for another CA, and the word “entity” is used to indicate the subject of a certificate (i.e. an entity is a user or a CA).

e) **certificate**
A certificate is a statement signed by a CA asserting that the CA has verified according to the certification policy that a given entity is the owner of a given public key (and, by implication, the corresponding private key) within a given time interval. The certificate also contains a pointer to the certification policy or policies under which the certificate was issued. There is no reason why the same public key should not be contained in multiple certificates issued by different CAs under different policies (as long as it meets the requirements of all the policies of course). However, this behaviour is sometimes deprecated.

f) **certification policy**
A certification policy of a CA describes the mechanisms by which a CA operates with respect to the entities that it certifies. The policy may describe such parameters as allowable encryption and signature algorithms, minimum key sizes, maximum length of certificate validity, maximum period between CRL updates, and allowable mechanisms for verifying the identity of the entity prior to certification. A policy will be named by an object identifier (OID), and the policy’s OID will be contained within all certificates issued under the policy.

g) **cross-certificate**
A cross-certificate is a statement signed by a CA that it believes another CA will implement a particular certification policy. Any entity trusting the cross-certifying CA will be able to verify certificates issued according to the cross-certified policy by the cross-certified CA. If the cross-certified CA has multiple

policies then the cross-certificate can specify which of these policies the CA is believed to implement.

h) **cross-certification policy**
A cross-certification policy of a CA describes what must be included in another CA's certification policy for a cross-certificate to be issuable. Note that even if another CA has an adequate certification policy, this does not mean that a cross-certificate has to be issued or can be assumed to have been issued. A cross-certification policy also describes the method a CA uses to verify the identity of another CA that is being cross-certified and the methods used to exchange public keys reliably.

i) **security domain**
A security domain is a logical grouping of CAs that conform to a common security policy, and their certified users. It can be as small as a single user with no CA, or it can be a single CA, a hierarchy of CAs or an arbitrary group of CAs.

j) **subordinate CA** and **superior CA**
If one CA issues a certificate for another CA, then the issuing CA is termed the superior CA, and the certified CA is termed the subordinate CA. The subordinate CA must implement the certification policy of the superior CA (though it can implement a stronger[1] policy if required). Note that there is no requirement that CAs are arranged in a hierarchy, these are just the terms to be used if they are.

k) **uncertified user**
An uncertified user is a user who has generated his/her own key pair and has not been certified by a CA.

l) **simple CA**
A simple CA is a security domain encompassing a single CA with a security policy, and a set of users all certified directly by that CA and all conforming to that security policy. All users of the CA must know the public key of the CA and must trust the CA.

m) **CA hierarchy**
A CA hierarchy is a security domain encompassing an arbitrary number of CAs and users. Exactly one CA shall be designated the root of the hierarchy, and this CA shall be superior to all other CAs within the domain (either directly or indirectly) and shall not be subordinate to any CA. All users within the domain must know the public key of the root CA and must trust the root CA. It is also sensible that users also trust their own CA and know its public key.

n) **out-of-band**
A "out-of-band" public key is one that is not verified using the trust model but is instead declared by a user to be "known with adequate confidence to be bound to a particular identity" (where 'adequate' is defined by the security requirements of the user). Often, such a key would have been obtained by insecure means (if so then it should be verified by a completely disjoint insecure means - the PGP key distribution mechanism using fingerprints provides a good example of this, though people do tend to distribute the key and the fingerprint by the same mechanism). Out-of-band public keys for users and for CAs are the seed on which the whole trust model is founded. Often out-of-band public keys will be distributed and stored as a "self-signed certificates" (which is simply a public key signed by its corresponding private key). From the point of view of the trust model, these are equivalent and self-signing does not add much protection. It is important to note that a public key infrastructure will need to define mechanisms to allow revocation of these keys.

o) **Trust**
Finally, it is worth re-iterating the definition of trust as specified in section 1 of the paper. Trust refers to the verifying user's confidence in the binding of identity and public key contained up in a certificate, and does not imply any inherent trustworthiness of the user that has been certified. The role of trust in the overall process of strong authentication of a user can be highlighted as follows:

- I *believe* that PKx is the public key of "CAx" (basic knowledge, obtained by out-of-band means).
- I *trust* "CAx" to issue correct certificates according to an acceptable policy.
- I can verify a certificate containing <PKy, USERy> signed with PKx
- Therefore I *believe* that PKy is the public key of USERy and that USERy is an accurate description

It is important to clarify the difference between certification and cross-certification in the context of this trust model, to avoid confusion with use of the same terms elsewhere. Certification corresponds to the issuing of a certificate within a security domain, whereas cross-certification corresponds to the issuing of a certificate between two different (non-overlapping) security domains.

There is no requirement that cross-certification be a two-way process - for two security domains, A and B, A can trust B while B does not trust A (presumably this happens because B provides more rigorous security than A).Therefore, the terms "one-way cross-certification" and

[1] Note that no method of comparing policies is defined. By "stronger" we mean that all certificates valid under the subordinate CA's policy must also be valid under the superior CA's policy. So if the superior CA requires 512 bit keys then the subordinate CA can keep the requirement or can strengthen it to require 1024 bit keys, but cannot relax the requirement and accept 256 bit keys. Similar rules can be derived for some other policy parameters such as CRL issuing frequency, but this cannot be extended generally.

"mutual cross-certification" can be used for clarification if required.

4 Principles of the trust model

In this section, we describe the principle components of the trust model, and their roles and responsibilities

- The trust model comprises CAs and users. CAs issue certificates and users do not. All CAs are equivalent in terms of what they are capable of doing (though, of course, some do it more securely than others). All CAs must publish a policy describing how they operate.
- All users have any number of private keys and corresponding public keys. The user may have generated their own key pair, or it may have been generated for them by either a certification authority or a third party registration authority.
- All CAs have a private key and a public key. The CA will normally have generated its own key pair, though it may have been generated by a superior certification authority.
- Any public key may be signed by zero or more certification authorities.
- Any user can explicitly declare belief in the identity of another user by obtaining an out-of-band copy of the user's public key.
- Any user can explicitly trust any CA by obtaining an out-of-band copy of the CA's public key and policy.
- Any CA can certify any user (provided the identity of the certified user is appropriate to the certification policy of the certifying CA) by publishing a signed copy of the user's public key.
- Any CA can certify any other CA (provided the identity and the certification policy of the subordinate CA is appropriate to the certification policy of the superior CA) by publishing a signed copy of the subordinate CA's public key.
- Any CA can cross-certify any other CA (provided the certification policy of the cross-certified CA is appropriate to the cross-certification policy of the cross-certifying CA) by publishing a signed copy of the CA's public key.

Every CA must have a certification policy that describes the operation of the CA with respect to the entities that it certifies (whether they are subordinate CAs or users). This policy will be published and a CA will usually expect its public key to be certified or cross-certified by other CAs who (a) find its policy acceptable and (b) believe it to implement the stated policy.

A CA will also have a cross-certification policy, which states the minimum policy that a CA requires another CA to implement in order to issue a cross-certificate. A cross-certificate means that any entity trusting the cross-certifying CA will be able to verify certificates signed by the cross-certified CA or by other subordinate CAs certified by the cross-certified CA.

Note that this does not extend to the ability to verify certificates signed by CAs cross-certified by the cross-certified CA (in other words, cross-certification is not transitive). This is because a cross-certificate signifies one CA accepting the ability of another CA to implement a certification policy, but does not make any statement at all about the other CA's cross-certification policy. This meets the requirement that the administrator of a security domain chooses which other domains are to be trusted by the users within his/her domain, and does not inherit other administrators' choices.

In general, there can be no single meaningful/simple picture of "the trust model". Strict hierarchies are easy to draw, but general mechanisms quickly become too complex to comprehend. Instead, each user can be considered to have their own view of the world with:

- themselves at the top of hierarchy
- at the second level comes everything for which they hold an out-of-band public key. For users who are part of one or more hierarchies, this will comprise the public keys of the root CA for each hierarchy. It will also comprise users and CAs whose public keys have been directly obtained.
- at the third level comes everything certified or cross-certified by the CAs at the second level
- at the fourth level comes everything certified by the CAs at the third level…. and so on

This ego-centric view does not require users to issue certificates. Instead, trust from a user towards a CA (i.e. 'up the hierarchy') is expressed in terms of the out-of-band public keys held by the user, and this trust is implicitly included at the beginning of all certification paths. 'Certification' of one user by another user is also expressed through out-of-band public keys, and user issued certificates are not required as this relationship is only relevant to the user who holds the out-of-band key.

5 Use of the trust model

In this section, we look at how a user uses the trust model to perform authentication. The general problem that trust models aim to solve can be worded as follows:

> *"Given some information about a user (usually a name or a public key) how do we determine which certificates and CRLs are needed to obtain an authenticated public key for that user, and who do we have to trust (what do we have to trust them to do)"*

The problem is worded in this way so that it is equally applicable to the problems of authenticating something signed by another user, and of discovering the public key for a user so that something can be encrypted for them. Therefore, it cannot assume that any other information is available (such as users providing full certification paths when sending PEM messages, or similar information being contained in directory services). If this information is available then it can obviously be used to optimise the performance of some applications using the trust model. However, the model should still be able to function without this information, even if less efficiently.

In order to achieve this goal, any user of the trust model will need some facility which is able to answer questions of the form “Give me all certificates for a particular entity” in order to search for certificates for users or CAs. This could be realised by a directory service, an Alta-Vista style WWW based search engine or any other lookup mechanism. It is up to implementations to provide sufficient indexing and caching to ensure performance is efficient.

There are two stages in the verification of a certificate using this and any other trust model. Firstly, through examination of digital signatures within a chain of certificates, you determine whether or not you believe the user's certificate to be accurate (i.e. unaltered). Having determined the integrity of the certificate, you then examine the policies of the CAs within the chain of certificates to determine whether the assurance that is provided as to the identity of the user is acceptable to your needs (i.e. that the name in the certificate is a good description of the actual owner of the certified public key). Note that the assurance required during the last stage may change depending on the context in which you are authenticating the user.

Unless a user has obtained the public key of another user by out-of-band means, the process of discovering an authenticated public key for that user involves constructing a certification path for that user. This involves retrieving the user’s certificates and recursively retrieving certificates for the issuers of those certificates until a trusted point is found (i.e. a point for which an out-of-band public key is held). The algorithm is a general spanning search of a directed graph, where the nodes are CAs and the edges are certificates and cross-certificates, and revoked certificates are edges to be avoided. However, the set of nodes, edges and avoids cannot be known in advance, and so the algorithm needs to build up the graph, starting from the target user’s CA and working backwards until a starting point is found that allows an acceptable path to be constructed or until all possibilities are exhausted. The algorithm for finding the certification path may be roughly stated as follows:

```
RETRIEVE all certificates issued for the
        user to be authenticated
INITIALISE graph with these nodes
DO forever
   IF graph contains a trusted CA then
      SET certification_path = path from
           trusted CA to user
      CHECK which certificates on
           certification_path are
           revoked
      IF some are revoked then
         REMOVE these edges from graph
      ELSE if certification_path
           contains zero or one cross
           certificates
         RETURN certification_path
   RETRIEVE all certificates issued for
        CAs in the graph
   IF none then
      QUIT
   ADD new nodes and links to graph
END
```

Once the verifying user has determined the certification path, this must then be checked for correctness. This means that the user verifies the public key in the first certificate using the out-of-band public key, then verifies the public key in the second certificate using the public key in the first certificate, and so on until the user’s public key is verified with the public key in the last CA certificate.

Next the verifying user checks the policy OIDs contained within the certificates. For a sequence of CAs within a hierarchical security domain, certificates issued by subordinate CAs must include the policy OID of the superior CA. For cross-certificates, certificates issued by the cross-certified CA must include the policy OID specified in the cross-certificate.

Finally, in order to determine a level of assurance that the user’s name within their certificate is an accurate description of the actual user, the verifier then needs to examine the policies of the CAs within the certification path. It is highly undesirable to make the verifying user examine every policy within a certification chain, and this can be avoided if certification paths are arranged such that they exhibit policy subordination. In this case, the user simply needs to examine the policy of the first CA in the certification path, and make sure that this policy is implemented by all CAs within the certification path (i.e. the user checks that the policy OID of the first CA is contained within all the certificates).

In general, the search for a certification path could return many different paths. The above algorithm returns only the first, and it is possible it will find a path using CAs implementing unacceptably weak policies and miss a path involving better policies. An even more simple-minded algorithm may even find a path including a revoked certificate and miss one that does not, leading the verifier to mistakenly believe the user cannot be authenticated. Therefore, it is often necessary to do a complete search for all certification paths and this search will, in general, be of exponential complexity. Implementations can do various

things to limit the complexity of the search (pruning the search tree) or to reduce the effort involved (local caching of certificates or certification paths), and there is the usual trade-off between the extent of the search for a certification path and the quality of the result obtained.

The practical scaling properties of this search are likely to be undesirable, and this is inevitable given that the design of the trust model has focused on generality rather than implementability. In the next section of the paper, we will describe some guidelines as to how we can modify the model slightly to give predictable performance while keeping its advantages.

6 Managing complexity

This is a general trust model, and as such it is perfectly possible to configure CAs such that there are too many certification paths between users and therefore implementation of the model becomes inefficient. This section offers some guidelines for CA administrators within the ICE-TEL project which aim to ensure the certification topology does not get overly complex and therefore that well-defined certification paths can always be constructed. We will start by proposing some strict rules, and then show how these rules can be relaxed without losing the simplifications they provide.

Restrictions to the model are required to ensure that the performance of applications implementing the trust model does not degrade unacceptably as the user population increases. One of the drawbacks of PGP is that its centralised key-server architecture does not scale gracefully beyond tens of thousands of users. In reality, applications could one day have to cope with hundreds of millions of users, each with many distinct certificates. So scalability is a crucial issue.

The principle reason for scaling problems in X.509 based certification systems is the difficulty in constructing certification paths in the presence of arbitrary certification. The verifying user needs to construct a chain of certificates that start at a point that is trusted by them and that ends with the user to be authenticated. PEM proposed a two part solution to this: firstly that there will be a single rooted hierarchy encompassing all security domains and that all users would know the public key of the root CA, and secondly that users sending signed messages would include with the message the full certification path from the root CA to themselves. This meant that the recipient of a PEM message is given all the information needed to verify the origin of the message (though the only policy information included in this was the name of the Policy CA, which is not sufficient). A user wanting to send an encrypted message to someone had a more difficult job determining their public key, but the certification path from the root CA to the user was well defined and so automated mechanisms could be devised.

In the ICE-TEL trust model, there is no single root CA certifying all security domains, and so this approach cannot be duplicated. Instead, the ICE-TEL model results in an arbitrary number of security domains, the domains being potentially interlinked by arbitrary cross-certification. In order to manage the resulting complexity, it is necessary to provide guidelines regarding the contents of a security domain and the way they may cross-certify each other.

Firstly, we can identity three types of security domain that will meet all likely requirements:

1. Hierarchies with two or more levels correspond to existing hierarchies such as described in RFC 1422. The root of the hierarchy will be a PCA which describes and advertises the minimum policy which subordinate CAs enforce and to which end entities adhere. This type of hierarchy is very suitable for large national or multinational companies with well-defined security needs. The security hierarchy can be organised on the basis of geographical distribution, company organisational distribution, distinct security needs of different groups of users or any other criteria.
2. Hierarchies with exactly one level. This corresponds to a simple CA issuing certificates for a collection of users. The CA must specify a security policy that it enforces. This arrangement is very suitable for small organisations or groups of users with a single set of security requirements. Of course, the direct trust CA may issue certificates for a very large number of users, making this model also suitable for a large company which does not wish to or need to replicate the certification process as required by a larger hierarchy.
3. Hierarchies with exactly zero levels. This corresponds to an uncertified user making its public key available out-of-band. There will be no policy statement involved. This arrangement, which corresponds closely to the PGP model, is very well suited for individual end users or for organisations with no overall security requirement where an individual user has a short term need for security.

We define a "**Trust Point**" (TP) as being the root of each of these types of hierarchy (the root being the PCA in the first case, the simple CA in the second case, and the uncertified user itself in the third case), and say that a security domain contains exactly one hierarchy. A user within a domain must trust the TP at the root of that security domain and must have received an out-of-band copy of the public key of that TP.

We can now specify cross-certification among security domains in terms of cross-certification among trust points, and use this to constrain the trust model in such a way as to ensure that implementations do not suffer from scaling

problems as the number of security domains increase, or as the size of any given security domain increases.

The following guidelines allow for the calculation of well defined certification paths:

1. security domains should be interlinked only by cross-certification among the TPs (this restriction reduces the number of places you have to look when building certification paths).
2. a user may only be a member of one security domain[2] (this restriction ensures that each user will have only one certification path).
3. when a user publishes their certificate (by a mechanism such as a white pages directory, a finger file or a WWW page) they also publish (in the same place) the certification path that starts with the TP at the root of their security domain and ends with themselves (this guideline means that all information that can be pre-calculated should be published along with the certificate, to give as much help as possible to the verifier)

Therefore, when user1 needs to authenticate a message signed by user2, there are four cases:

1. user1holds an out-of-band public key for user2, and so can verify the message immediately.
2. user1 and user2 are in the same security domain. User1 will retrieve the certificate and certification path for user2, and will see that user 1 already trusts the TP of user2. Therefore, user1 will be able to authenticate the message immediately.
3. user1 and user2 are in different security domains, domain1 and domain2, where the two domains have cross-certified each other. User1 will retrieve the certificate and certification path for user2, and will see that user1 does not trust user2's TP. User1 will retrieve the list of cross-certificates issued by the TP at the root of domain1, and will extract the cross-certificate. User1 will add this cross-certificate to the beginning of users2's certification path and will then be able to authenticate the message.
4. user1 and user2 are in different security domains, domain1 and domain2, where the two domains have not cross-certified each other. User1 will retrieve the certificate and certification path for user2, and will see that user1 does not trust user2's TP. User1 will retrieve the list of cross-certificates issued by the TP at the root of domain1, and will find that the required cross-certificate has not been issued. User1 will therefore be unable to authenticate the message from user2.

[2] A person can be a member of multiple security domains if he/she possesses multiple certificates issued by different CAs within different security domains, but the trust model is not aware that these are the same user.

If user1 is trying to retrieve a public key for user2 in order to send an encrypted message then the same three cases and solutions apply.

It is important to note that, as the ICE-TEL trust model has prohibited transitive cross-certification, the process of constructing the certification path across domain boundaries is reduced to only the simple question of whether the TP of user1 has cross-certified the TP of user2. It is never necessary to search for one or more intermediate CAs that comprise a chain. Therefore, the time taken is independent of both the number of security domains and the size of any security domain.

In fact, the cross-certification rules can be relaxed slightly without sacrificing performance. As users are still allowed to obtain out-of-band public keys of other CAs they wish to trust, a certification path can be constructed if a user can find a cross-certificate for the TP of user2 issued by any of the CAs that they trust. This means that it is still useful for CAs within a hierarchy to issue cross-certificates, and the time taken is still independent of the number of security domains and the size. Furthermore, a path can also be constructed if user1 can find a cross certificate for any of the CAs in the certification path between the TP of domain2 and user2, so it is still useful to issue cross-certificates for CAs within a hierarchy. If this is allowed then the time is now dependent on the depth of hierarchies within security domains, but still not on the number of other domains.

Finally, the requirement that users belong to only one security domain can be relaxed if the method of publishing certification paths allows multiple paths to be published. The arrangement of CAs is required to be sufficiently well-defined that all users know which security domains they are in, and each domain is still required to have a trusted point at its root. Users now publish a number of certification paths, from the TP of each security domain to themselves.

The scalability of any implementation of the model also depends on the scalability of other mechanisms involved in the overall public key infrastructure, namely: for users to advertise their certificates and certification paths; for a CA to advertise the cross-certificates that it has issued; and for distribution of CRLs. Methods of certificate and CRL distribution are not covered in this paper, and will be covered in other parts of the ICE-TEL project and the work of the IETF PKIX working group.

This simplified trust model may look, at first glance, to be very similar to RFC1422. However, it provides additional flexibility by abolishing the need for a single root and by allowing TPs to cross-certify only CAs with acceptable policies rather than relying only on only the PCA name to determine whether the authentication is acceptable. It also can handle extra cross-certification without sacrificing performance.

7 Examples

The following certification infrastructure will be used to demonstrate a few points. To summarise, it comprises four separate security domains (UCL, SALFORD, ICE-TEL and SNI), each inter-linked by a limited amount of cross-certification. There is also some overlap between the domains.

- arrows show cross-certification (and are directional)
- upper case shows CAs, lower case shows users

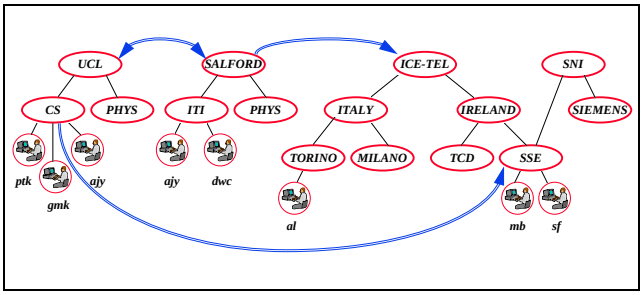

Figure 1 - Example certification infrastructure

7.1 Certification policy statements

The CAs at the roots of the example hierarchies will be assumed to have the following (somewhat contrived and incomplete) certification policies:

UCL CA Policy: The UCL CA will certify entities within the UCL.AC.UK name-space; and CRLs will be issued at least once per week. Subordinate CAs must also implement this policy.

SALFORD CA Policy: The SALFORD CA will certify entities within the SALFORD.AC.UK name-space; and will issue CRLs once per week. Subordinate CAs must also implement this policy.

ICE-TEL CA Policy: The ICE-TEL CA will sign RSA keys of 512 bits or longer; certificates will be valid not beyond 30th November 1997; and CRLs will be issued every day. Subordinate CAs must also implement this policy.

SNI CA Policy: The SNI CA will sign RSA or DSA keys of 1024 bits or longer; certificates will be valid for one year; and CRLs will be issued every day. Subordinate CAs must also implement this policy.

In all cases, a subordinate CA must implement at least the same policy as the superior CA, and this 'policy subordination' is omitted from future descriptions for reasons of compactness. Note that it is possible for a CA to simultaneously implement both the ICE-TEL policy and the SNI policy until 30th November 1996, but not afterwards.

Subordinate CAs within the hierarchies will be assumed to have the following certification policies:

CS The CS CA will certify entities within the CS.UCL.AC.UK name-space and will issue CRLs weekly.

ITI: The ITI CA will certify entities within the ITI.SALFORD.AC.UK name-space and will issue CRLs twice weekly.

IRELAND: The IRELAND CA will certify entities within the c=IRELAND name-space that implement the ICE-TEL policy, and will issue CRLs daily.

ITALY: The ITALY CA will certify entities within the c=ITALY name-space that implement the ICE-TEL policy and will issue CRLs daily.

SSE: The SSE CA will certify entities within the SSE.IE name-space; and will sign RSA keys of 1024 bits or longer. Certificates will be valid either for a year or until 30/11/97, depending in user requirements (an SNI certificate will be issued in the former case, an ICE-TEL certificate in the latter case, a combined certificate will be issued before 30/11/96). CRLs will be issued daily.

TORINO: The TORINO CA will certify entities within the TORINO.IT name-space according to the ICE-TEL policy and will issue CRLs daily.

Note that the SSE CA must implement two policies. If the policies are complementary then this is no problem, and so before 30/11/96 the SSE CA can issue a single certificate containing the OIDs of both the ICE-TEL policy and the SNI policy. If policies conflict then either a different CA could be run for each policy, or a single CA could issue users with multiple certificates saying which of the policies they had been certified under. Therefore SSE could issue users with a certificate, valid until 30/11/97, containing the OID of the ICE-TEL policy; and also with a certificate, valid for a year, containing the OID of the SNI policy. After 30/11/97, people trusting only the ICE-TEL CA would be unable to verify any signatures made by SSE users, which would be exactly what would be intended.

7.2 Cross-certification policy statements

Three CAs in the example are shown as having issued cross-certificates. The (very contrived) cross-certification policies of the CAs are assumed to be as follows:

UCL: The UCL CA may cross-certify any CA in the UK academic community.

SALFORD: The SALFORD CA may cross-certify any CA that issues weekly CRLs.

CS: The CS CA may cross-certify any CA including name subordination among its policies.

The example includes the following four cross-certificates:

UCL cross-certifies **SALFORD** and trusts SALFORD to certify entities within the SALFORD.AC.UK name-space.

SALFORD cross-certifies **UCL** and trusts UCL to certify entities within the UCL.AC.UK name-space.

SALFORD cross-certifies **ICE-TEL** and trusts ICE-TEL to certify entities implementing the ICE-TEL policy.

CS cross-certifies **SSE** and trusts SSE to certify entities within the SSE.IE name-space.

Note that the SSE CA's certification policy specifies a number of things, and the CS CA is cross-certifying that it believes it to do one of them.

7.3 Worked examples

The following examples assume that each user directly trusts only their own CA and the CA at the root of the hierarchies that they belong to.

Example 1, showing non-uniqueness of names and authentication within one security domain. dwc is authenticating a message from ajy, given just a name and a public key. There are two ajys, and dwc does not immediately know which is the correct one.

i) dwc retrieves all certificates for ajy
ii) answer is {ITI→ajy, CS→ajy} which correspond to two logically different ajy's (dwc does not know if they relate to the same physical person)
iii) The public key held by dwc matches the key in the ITI→ajy certificate, so this ajy is indicated as the one being authenticated.
iv) dwc knows ITI's public key and can therefore immediately verify the certificate chain
v) ajy's credentials conform to ITI's certification policy
vi) dwc trusts ITI to implement its certification policy
vii) therefore dwc believes that it really is "ajy from ITI in Salford University"
viii) message signature verifies, therefore authenticated message really was signed by "ajy from ITI in Salford University"

Further searching would have revealed two more certification paths:

SALFORD→ITI→ajy
SALFORD→UCL→CS→ajy

The first of these is also a valid path, and the authentication would have occurred under SALFORD's certification policy. The second is also a valid path for the "other" ajy, so dwc can converse securely with both and can distinguish them.

If dwc had only the name and not the public key, then the ITI→ajy certificate could still have been selected by checking which of the two certified public keys was the correct one for message verification. However, if dwc had been attempting to obtain a key for ajy for the purposes of encrypting a message to be sent to ajy then it would have been necessary to derive verifiable certification paths for both ajys and then allow dwc to choose which was the correct one, given as much policy information as possible to allow dwc to disambiguate the choices. If non-unique name forms are used for certificate selection then this manual confirmation step will always be necessary.

Using the simplified model, ajy makes the following information available as part of the message:

Certificate: ITI→ajy
Certification path: SALFORD→ITI→ajy

dwc knows the public key and policy of the SALFORD CA and so can verify the certification path, and can use the policy of the CA to determine a level of assurance.

Example 2, showing authentication across security domains. dwc is obtaining an authenticated public key for al

i) dwc retrieves all certificates for al
ii) answer is {TORINO→al}
iii) dwc retrieves all certificates for TORINO
iv) answer is {ITALY→TORINO}
v) dwc retrieves all certificates for ITALY
vi) answer is {ICE-TEL→ITALY}
vii) dwc retrieves all certificates for ICE-TEL
viii) answer is {SALFORD→ICE-TEL}
ix) dwc knows SALFORD's public key and can therefore verify the certificate chain
x) al conforms to TORINO's certification policy
xi) TORINO→al conforms to ITALY's certification policy
xii) ITALY→TORINO→al conforms to ICE-TEL's certification policy
xiii) ICE-TEL→ITALY→TORINO→al does not conform to SALFORD's certification policy
xiv) ICE-TEL→ITALY→TORINO→al does conform to the SALFORD→ICE-TEL cross-certificate
xv) dwc trusts SALFORD's cross-certification policy
xvi) therefore dwc believes that it really is al's public key

Using the simplified model, al makes the following information available:

Certificate: TORINO→al

Certification path: ICE-TEL→ITALY→TORINO→al

dwc does not know the public key of the ICE-TEL CA, but retrieves the list of cross-certificates issued by the SALFORD CA and sees that the SALFORD CA has cross-certified the ICE-TEL CA. dwc knows the public key of the SALFORD CA and so can construct a certification path SALFORD→ICE-TEL→ITALY→TORINO→al which can be verified. dwc can also find which policy was cross-certified, and use that to determine a level of assurance.

Example 3, showing transitive cross-certification. ptk is trying to obtain an authenticated public key for al.

i) ptk retrieves all certificates for al
ii) answer is {TORINO→al}
iii) ptk retrieves all certificates for TORINO
iv) answer is {ITALY→TORINO}
v) ptk retrieves all certificates for ITALY
vi) answer is {ICE-TEL→ITALY}
vii) ptk retrieves all certificates for ICE-TEL
viii) answer is {SALFORD→ICE-TEL}
ix) ptk retrieves all certificates for SALFORD
x) answer is {UCL→SALFORD}
xi) ptk knows UCL's public key and can therefore verify the certificate chain
xii) al conforms to TORINO's certification policy
xiii) TORINO→al conforms to ITALY's certification policy
xiv) ITALY→TORINO→al conforms to ICE-TEL's certification policy
xv) ICE-TEL→ITALY→TORINO→al does not conform to SALFORD's certification policy
xvi) ICE-TEL→ITALY→TORINO→al conforms to the SALFORD→ICE-TEL cross-certificate
xvii) SALFORD→ICE-TEL→ITALY→TORINO→al does not conform to UCL's certification policy
xviii) SALFORD→ICE-TEL→ITALY→TORINO→al does not conform to the UCL→SALFORD cross-certificate
xix) therefore PTK does not believe that it really is al, because UCL's cross-certificate does not permit the rest of the certificate chain.

Using the simplified model, al makes the following information available:

Certificate: TORINO→al

Certification path: ICE-TEL→ITALY→TORINO→al

ptk does not know the public key of the ICE-TEL CA, and so retrieves the list of cross-certificates issued by the UCL CA. ptk sees that the UCL CA has not cross-certified the ICE-TEL CA or any other CA within the certification path. So ptk retrieves the list of cross-certificates issued by the CS CA. The CS CA has also not cross-certified the ICE-TEL CA or any other CA within the certification path, and so ptk is unable to authenticate al's certificate.

Example 4, showing cross-certification within hierarchies and overlapping security domains. ptk is obtaining an authenticated public key for sf

i) ptk retrieves all certificates for sf
ii) answer is {SSE→sf}. Note that two certificates may be returned if SSE is issuing separate certificates for the ICE-TEL policy and the SNI policy.
iii) ptk retrieves all certificates for SSE
iv) answer is {IRELAND→SSE, CS→SSE, SNI→SSE} which all correspond to the same SSE (this is easily recognisable if the same CA public key is certified in all three certificates).
v) ptk knows CS's public key and can therefore verify the certificate chain
vi) sf conforms to SSE's certification policy
vii) SSE→sf does not conform to CS's certification policy,
viii) SSE→sf does conform to the CS→SSE cross-certificate
ix) ptk trusts CS's cross-certification policy
x) therefore ptk believes that it really is sf

Further searching would have revealed another certification path:

UCL→SALFORD→ICE-TEL→IRELAND→SSE→sf

This is an invalid path as it involves two cross-certificates. Mechanical verification would have spotted this because SALFORD→ICE-TEL is not permitted by the cross-certified policy specified in the UCL→SALFORD cross-certificate.

Using the simplified model, sf makes the following information available as part of the message:

Certificate: SSE→sf

Certification path: ICE-TEL→IRELAND→SSE→sf

Certification path: SNI→SSE→sf

ptk does not know the public key of the ICE-TEL CA or the SNI CA, and so retrieves the list of cross-certificates issued by the UCL CA. ptk sees that the UCL CA has not cross-certified the ICE-TEL CA, the SNI CA or any other CA within the certification path. So ptk retrieves the list of cross-certificates issued by the CS CA. The CS CA has also not cross-certified the ICE-TEL CA or the SNI CA, but has cross-certified the SSE CA. ptk knows the public key of the CS CA and so can construct a certification path CS→SSE→sf which can be verified. ptk can also find which policy was cross-certified, and use that to determine a level of assurance.

8 Organic growth

In order for a security infrastructure to become widely used, it is important that the level of effort required from users of the infrastructure is proportional to the use and benefit they expect to obtain from the infrastructure. In particular, small scale users with occasional security needs should have a small start-up cost. The three different types of security domain introduced by this trust model ensure that this requirement is met.

For users with small scale security requirements, the model allows them to generate a key pair for themselves and exchange their public key with users with whom they want to communicate securely. Companies with medium security requirements can create a CA to certify users within the company and can cross-certify other companies they deal with as the need arises. Companies with large security requirements can create a hierarchy of different CAs with different policies and with different cross-certificates for each part of the hierarchy.

However, it is also important that the model allows the security requirements of a user or an organisation to change over time, and that these changes do not require disproportionate amounts of work. Migration between the three types of security domain (users, CAs, hierarchies) should be well defined. In particular, the following five transitions can be identified:

1. One or more uncertified users merge to form a simple CA: the users would previously have had self-issued key-pairs, and these would all be signed by the newly created CA according to its newly devised certification policy. The public key of the new CA would be distributed to the users using the same mechanism that a CA would normally use to distribute its key to a new user, and the users would retain this as an out-of-band key. If the users previously held out-of-band public keys for other users or CAs then these can be retained. The keys of the CAs could also be signed by the CA according to its cross-certification policy, thereby becoming cross-certificates for all users of the CA.
2. One or more CAs merge to form a hierarchy (or hierarchies merge): the public keys of the CAs would be signed by the newly created root CA of the new hierarchy, and the public key of the new root CA distributed to the original CAs using the same mechanism that a CA would normally use to distribute its key to a new subordinate. The CAs would then send the new root public key (as a signed email message) to their users and subordinates, and users would retain this as an out-of-band key. The new root CA would devise and advertise a certification policy and a cross-certification policy according to the weakest policies of the original CAs. Any cross-certificates issued by the original CAs can be retained by those CAs, or they can be re-signed by the new root CA as long as the cross-certification policy permits.
3. A hierarchy divides into a collection of CAs (or other hierarchies): each CA subordinate to the root CA of the dividing hierarchy would pass its public key down the new hierarchy via its subordinate CAs (the key can be sent as a secure email message, since each stage down the hierarchy would know the public key of the immediately superior CA). Each of these CAs becomes the root of a new hierarchy (or becomes an isolated CA if the original hierarchy was only two deep), and so is retained by users as an out-of-band key. The new root CAs may choose to cross-certify the CAs that were previously subordinate to the defunct root CA, and they may also choose to sign the public keys in any cross-certificates signed by the former root. Users would discard the public key of the former root CA and any certificates signed with that key.
4. A CA divides into a collection of uncertified users: each user would discard the public key of the CA and their certificate signed by that CA. The users may choose to add the public keys of other user certificates of the original CA to their out-of-band keys, and may also add the public keys in the cross-certificates issued by the CA to their out-of-band public keys.
5. A CA subdivides into a two or more CAs: one or more new CAs will be created (each with a newly devised certification policy and cross-certification policy) which will sign the public keys of existing users (or can, of course, issue the users with new key pairs) and will securely pass its public key to the users (it can be sent as a signed email message signed by the old CA). The users do not need to discard the public key or certificate of the old CA.

9 Conclusions

The ICE-TEL trust model proposed here mixes the “hierarchical trust” and “web of trust” models to form a “web of hierarchies” model. It has many advantages over other trust models in common use:

1. The ICE-TEL trust model provides a scaleable deployment model, where communication between two simple users requires no infrastructure at all, communication between two organisations with simple security requirements requires a simple infrastructure (just a CA each), communication between arbitrarily complex organisations requires a larger infrastructure, and changing requirements are easily accommodated. Compare this with PEM, which requires a significant infrastructure to be in place before two organisations can start communication, and does not relax these requirements for small scale users; and with PGP which

allows users to communicate with little or no infrastructure, but does not significantly increase the infrastructure available for users with complex requirements.

2. The ICE-TEL trust model makes formal use of CA policy, and specifies use of the X.509v3 extensions relating to policy. The use of policy subordination within security domains makes the model more flexible than the PEM trust model, which uses name subordination to constrain what subordinate CAs can certify. All CAs are required to publish a policy (compare with PEM where only designated Policy CAs publish policies, and with PGP where no policies are defined) and this allows all CAs to be treated equally (any CA at any level of a hierarchy can cross-certify and can be cross-certified, though this can lead to excessive complexity if overdone). Users do not have policies and so are not allowed to issue certificates.

Applications implementing the trust model are currently being prepared by ICE-TEL technology provider partners, and will soon be in use by European CA service providers and pilot sites.

Acknowledgements

The authors would like to acknowledge the comments and reviews of Sead Muftic of COST, Sweden and Stephen Farrell of SSE, Ireland. The support of the entire ICE-TEL project team is also acknowledged.

Contact information

The authors can be contacted by electronic mail as follows:

- Andrew Young <A.J.Young@iti.salford.ac.uk>
- Nada Kapidzic Cicovic <nada@cost.se>
- David Chadwick <D.W.Chadwick@iti.salford.ac.uk>

References

[1] CCITT, Recommendation X.500: *The Directory - Overview of Concepts, Models and Services*, 1993
CCITT, Recommendation X.509: *The Directory Authentication Framework*, 1993

[2] Linn, J, *Privacy Enhancement for Internet Electronic Mail: Part 1: Message Encryption and Authentication Procedures, RFC 1421*, DEC, February 1993
Kent, S., *Privacy Enhancement for Internet Electronic Mail: Part 2: Certificate Based Key Management, RFC 1422*, BBN, February 1993
Balenson, D., *Privacy Enhancement for Internet Electronic Mail: Part 3: Algorithms, Modes, and Identifiers, RFC 1423*, TIS, February 1993
Kaliski, B., *Privacy Enhancement for Internet Electronic Mail: Part 4: Notary, Co-Issuer, CRL-Storing and CRL-Retrieving Services, RFC 1424*, RSA Laboratories, February 1993

[3] Zimmermann, P., *Pretty Good Privacy: Public Key Encryption for the Masses*, PGP User's Guide, May 1994

[4] ICE-TEL home page http://www.darmstadt.gmd.de/ice-tel

[5] ISO/IEC JTC 1/SC 21/WG4 and ITU-T Q 15/7, *Draft Amandment DAM 1 to ISO/IEC 9594-8 on Certificate Extensions*, June 1996